Combined Task and Motion Planning under Partial Observability: An Optimization-Based Approach

Camille Phiquepal **Marc Toussaint**
Machine Learning & Robotic Lab, University of Stuttgart
{firstname.surname}@ipvs.uni-stuttgart.de

***Abstract*—We consider Task and Motion Planning (TAMP) problems for object manipulation under partial observability. More specifically, we aim at defining long-term policies for manipulation problems in environments where a robot has to explore its environment to complete its task (for example, look inside a box to check to see if an object is inside). Our Partially-Observable-Logic-Geometric-Programming approach (POLGP) is a trajectory-optimization-based approach of TAMP and builds on the Logic-Geometric-Programming framework (LGP) presented in prior work [1, 2]. To enable the robot to explore its environment, we add "perceptual actions" (for example *Look*) to the robot's classic actions (*Pick, Place, etc.*) used for manipulation problems. The perceptual actions aim at placing the robot sensor where it can gain information. To represent partial observability, we enable the planner to reason about the agent belief-state on both a symbolic and geometric level. Perceptual actions lead to a branching point in the policy. A TAMP policy is, therefore, not a sequence of actions but a tree of actions. To handle this specificity, we introduce the notion of "trajectory-tree optimization" for optimizing trajectories across branching points. To our knowledge, this is the first TAMP approach that computes long-term policies under partial observability and, hence, computes trajectories that are not sequential but arborescent.**

I. Introduction

As an example, we consider a robot equipped with a vision sensor and having to fetch an object placed in a box. The robot is surrounded by several boxes and doesn't know which one contains the object to fetch. To perform its task, the robot has to "take a look" inside the boxes (see Figure 1). To do so, the robot has to move its sensor in order to have the target location within its field of view. If some boxes are too far away, the robot must grasp them to bring them closer and, simultaneously, move its sensor. Some boxes may be closed. The robot would have to open them first. Our current work aims at capturing this kind of robot behavior.

Most current TAMP research on object manipulation assumes full observability [1, 2, 3]. We believe that partial observability is pervasive in many real world situations. When the environment is cluttered, object recognition may fail because objects may be hidden or partially hidden. If objects are inside containers (objects inside a cabinet, a box, etc.), the objects may not be visible at all at first. To solve this, we enrich the action space with perceptual actions, like *Look* : The agent places its sensor (or move existing objects) so that the location to observe is in the sensor field of view. Object recognition is, then, triggered and outputs a binary observation (whether the object has been seen or not). The observation is used to update the belief state. The geometric implementation of those perceptual actions is closely related to robotic sub-fields traditionally called "Sensor Placement" or "Active Perception". In [4], Lozano-Perez and Kaebling describe an approach which explicitly models partial observability and, also uses a *Look* action to gain information. This approach (Hierarchical Planning in the Now) interweaves planning with execution (in the now). The system plans sequences of actions by approximating the system dynamics (results of actions and observations). Replanning is triggered once the robot ends up in a state not covered by the plan. Our approach aims at planning a full policy from the starting state to the final state.

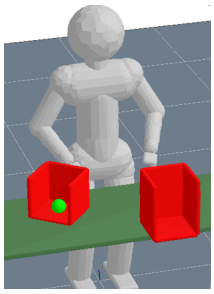

Fig. 1. The robot doesn't know at first which box contains the ball

II. Working hypothesis from a POMDP perspective

1) Partial Observability and the Observation model: The POLGP approach aims at handling a "discrete" or "categorical" partial observability of the environment. For example, the robot doesn't know beforehand which box the object is in. This method doesn't intend to cope with a more "continuous" partial observability like a residual uncertainty of the objects' positions. We believe that this latter kind of uncertainty can be managed at execution time with a controller able to adapt to small changes in the environment.

2) Uncertainty: Our current research assumes that the action model is deterministic. However, actions may fail at execution time.

In future work, we intend to explore ways to handle this uncertainty. One common contingency strategy would be to reattempt an action that has failed i.e. adding cycles in the policy. The definition of the motion planning problem for contingency actions is, however, difficult. Indeed, when an action fails (for instance, if a grasped object falls), the system

potentially ends up in an infinity of very different kinematic states: the grasped object may fall upside down, strongly impacting how to re-grasp it. One solution could be to defer the motion planning of the contingency actions / cycles to execution time.

3) Conclusion: The POLGP approach can be considered a particular case of a Non-Stochastic-Continuous-State-POMDP. Although it may seem over-optimistic compared to a full POMDP approach, we think that these are important working hypotheses for keeping the problem tractable.

III. Arborescent nature of the policy

Perceptual actions have a structural impact on the planning process because they lead to a branching point. A long-term policy must have a plan which reacts to each percept. For example: "take a look inside the first box; IF the object is there, then pick it up, ELSE, take a look inside the second box, etc...". We use a Monte-Carlo AND / OR tree search for optimizing such arborescent policies.

The arborescent nature of the policy also has an important impact on a geometric level. The geometric instantiation of a policy is not a trajectory but a trajectory-tree. Our current research formulates trajectory-tree optimization as a generalization of a trajectory optimization.

IV. Trajectory-tree optimization

The Partially-Observable-Logic-Geometric-Programming approach builds on the Logic-Geometric-Programming framework (LGP); presented in prior work [1, 2]. Under this framework, each agent state-action on a symbolic level is "grounded" on a geometric level: it specifies costs and constraints on the trajectory-tree for a given time interval.

Therefore, a policy defines a constrained trajectory optimization problem. It is solved using non-linear mathematical programming (NLP) techniques to efficiently find smooth (locally) optimal paths.

V. Interaction between the symbolic and geometric level

The policy optimization is performed by alternating a symbolic search and trajectory-tree optimization. When the planning process is started, the planner only uses heuristic values for the expected cost of each action. The AND / OR Monte-Carlo-Tree-Search is performed until a "candidate policy" is found. Then, the trajectory optimization is performed. If the trajectory optimization succeeds, the policy is stored in a set of solutions. Otherwise, the part of the tree whose trajectory optimization failed is pruned. In both cases, the symbolic search is pursued to generate new candidate policies that are then optimized. This iterative process is continued until some termination criterion is reached (time limit, trajectory cost).

With our current implementation, the additional complexity implied by partial observability is linear with respect to the size of the belief state. This raises a particular challenge for the scalability: for a robot having to achieve manipulations on N objects, each one being at M potential locations, the planning time is expected to be M^N bigger compared to the observable case (positions initially known). In future work, we intend to discuss the complexity challenge, as well as to describe the conditions under which this approach can guarantee completeness and optimality.

VI. Example

Consider the set-up of Figure 1. The robot has 4 actions: look inside a container, grasp an object, place an object on the table and "home": go back to the start position. The goal condition is that the robot has seen the ball, that the two containers are on the table, and that the robot is at its start position. Figure 2 shows two candidate policies. The policy (a) is found first because it appears more optimal. However, the trajectory-tree optimization indicates that this policy is infeasible: since the taller container (container_1) is far away from the robot, there is no reachable point of view that allows the robot to look inside, which causes the trajectory optimization to fail. The robot has to grasp the container first to bring it closer (policy (b)). For viewing the optimized trajectories and their executions, we refer the reader to the following video: https://youtu.be/eZMLxteKrj8.

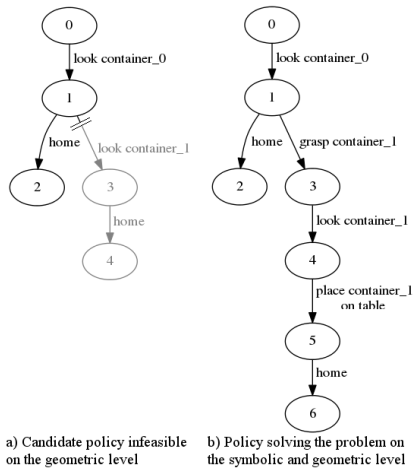

Fig. 2. Example of policies, only the second policy (b) is a solution. The first option (a), is feasible symbolically but not geometrically

References

[1] M. Toussaint and M. Lopes, Multi-Bound Tree Search for Logic-Geometric Programming in Cooperative Manipulation Domains. Accepted at ICRA 2017.

[2] M. Toussaint, Logic-Geometric Programming: An Optimization-Based Approach to Combined Task and Motion Planning. In Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI 2015), 2015.

[3] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, Incremental Task and Motion Planning: A Constraint-Based Approach, in Robotics: Science and Systems, 2016

[4] Leslie Pack Kaelbling, Tomas Lozano-Perez. Integrated Task and Motion Planning in Belief Space, International Journal of Robotics Research, 2013